一种面向 PDF 文本内容审查的高效多模式匹配算法 *

刘邦国，陈庆春，类先富

(西南交通大学 信息科学与技术学院，成都 611756)

摘　要：多模式匹配算法是网络入侵检测和内容过滤的核心算法。针对 Wu-Manber 多模式匹配算法所存在的匹配效率低、跳转距离较小的问题，结合 PDF 文本内容的编码规则，提出了一种适用于中文 PDF 文本内容审查的 Wu-Manber 改进算法。该算法使用布隆过滤器提取模式串关键信息，同时结合双重哈希和 PDF 文本编码规则，减少了无谓的匹配次数，加大了跳转幅度，从而提升了 PDF 文本的匹配性能。实验结果表明，这种改进算法在 PDF 文本审查中的匹配速率有较大提升，尤其当最短模式串较长且模式串规模较大时速度可以提升一倍以上。

关键词：多模式匹配；Wu-Manber 算法；PDF 文本编码；布隆过滤器

中图分类号：TP301.6　　doi: 10.19734/j.issn.1001-3695.2018.11.0868

Efficient multi-pattern matching algorithm for PDF content search

Liu Bangguo, Chen Qingchun, Lei Xianfu

(*Southwest Jiaotong University, School of Information Science & Technology, Chengdu* 611756, *China*)

Abstract: Multi-pattern matching plays an important role in network intrusion detection and content filtering. To solve the deficiency of Wu-Manber multi-pattern matching algorithm in terms of the achieved matching efficiency and jump distance, propose an improved Wu-Manber algorithm for Chinese PDF document content revciew on the basis of the coding formats of PDF document content. By employing the Bloom filter to extract the curcial information of the pattern string, and exploiting the double hash and PDF document encoding rules, it is shown that the proposed improved algorithm is able to reduce the number of unnecessary matches and increase the jump distance, thus improving the matching efficiency for the content retrieval of PDF document. The practical experimental results confirms the improved matching efficiency for PDF document. When the shortest mode string is long and the mode string size is large, the matching efficiency can be even doubled.

Key words: multi-pattern matching; Wu-Manber algorithm; PDF text encoding; bloom filter

0　引言

多模式匹配算法用于在文本中寻找出特定模式集中所有出现的模式串，在网络入侵检测和防御[1, 2]、病毒扫描、垃圾邮件检测、网络带宽和服务质量评估[3]等网络信息安全领域有十分广泛的应用。此外，多模式匹配在文本检索、搜索引擎和生物基因序列分析[4]等多个领域也有相关的应用。

随着互联网流量激增，本文已经进入信息和大数据时代，但随之也伴随着出现一些含有不良信息的网络文件，这些文件多以动态网页、静态文件（如 PDF 文档）等形式出现。便携式文档格式 PDF（portable document format）是一种标签命令式的文件格式，文件中可以包含超链接、表格、文字、图片等多种内容。目前网络中越来越多的信息流倾向于采用 PDF 文档来保存，对 PDF 文件的文本内容进行解析并对其中的敏感内容进行管控成为一个愈发重要的课题[5]。尤其当模式串规模较大时，如何在 PDF 文档内容的匹配检索过程中减少计算资源和存储资源消耗，提升和改善搜索和匹配过程效率，这是本文的主要研究背景和出发点。

本文在经典的 Wu-Manber[6]多模式匹配算法的基础上，结合中文 PDF 内容流的编码规则，提出了一种面向中文 PDF 文本内容审查的 Wu-Manber 改进算法。该算法使用布隆过滤器提取模式串关键信息，同时结合双重哈希和 PDF 文本编码规则，减少了无谓的匹配次数，加大了跳转幅度，从而提升了 PDF 文本的匹配性能。

1　PDF 文件结构及编码规则

一个完整的 PDF 文件从结构上分为四部分，即文件头(head)、交叉引用表(xref)、文件体(body)和文件尾(trailer)。其中文件体(body)是 PDF 文件的主要部分，由很多串行化的间接对象构成，它们共同组成了 PDF 文件的实际内容，如页面、字体、图形等。此外，文件体的这些间接对象之间有层次等级关系，需要各个对象之间层层调用来引出具体内容，形成了如图 1 所示的树型结构，其中页面(page)的 Contents 属性中 stream 与 endstream 之间的数据流便是被压缩后的文本内容。

PDF 的文本内容以十六进制形式表示，位于 stream 流中操作符 TJ/Tj 的前面，形式如<6C2985F61A09>TJ，其编码方式一般是 Unicode、GBK 或 CID 格式。由于中文字符是宽字符且数目较大，为了减少 PDF 文件大小，一般采用 CID 编码。此外，不同的 PDF 文件中文文本的 CID 编码各不相同，所以每个中文 PDF 文件存在映射关系表 cMap，用于储存中文文本的 Unicode 编码数据与 CID 编码数据的对应关系。例如，“模式”的 Unicode 编码数据为 6A215F0F，在某 PDF 文件的间接对象中对应的 CID 编码数据为<1F4E143C>TJ，

收稿日期：2018-11-16；**修回日期：**2019-01-10　**基金项目：**国家自然科学基金资助项目（61771406）

作者简介：刘邦国（1993-），男，山西吕梁人，硕士研究生，主要研究方向为模式匹配、自然语言处理（bangguoliu@163.com）；陈庆春（1973-），男，教授，博士，主要研究方向为信息编码；类先富（1981-），男，副教授，博士，主要研究方向为下一代通信技术.

从中可以看出，一个汉字对应的 CID 编码是由四个字符组成的。

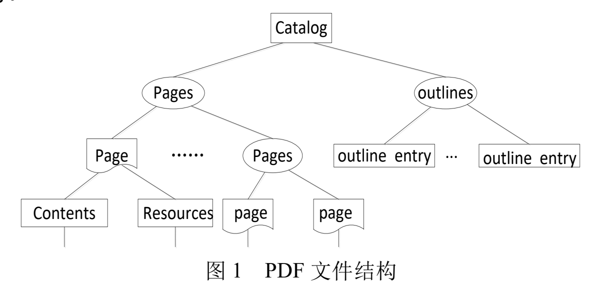


图 1　PDF 文件结构

Fig. 1　File structure of PDF

2　多模式匹配算法研究现状

面向 PDF 文本内容审查的多模式匹配是经典多模式匹配算法的一种特殊应用。目前多模式匹配算法的研究大致可以分为三类，以下作具体的阐述：

a）基于有限状态机匹配算法。这类算法的典型代表是 AC[7]算法。AC 算法具有匹配性能较稳定、受模式串长度和特征影响小等优点。但是 AC 算法建立 Trie 状态树的时间和空间复杂度随模式串集合线性增长，当模式串集合规模过大时，容易出现“内存墙”的问题，即消耗大量内存导致机器宕机，而且一旦增加或删除模式串都需要重新建立 Trie 状态树。针对以上问题，文献[8]提出了一种改进算法 HashTrie 来优化空间利用率，通过使用递归哈希函数将模式串的信息存储在位图中，从而用位图来取代状态转移表从而减少内存开销。这种算法虽然相比 AC 算法节省了 99.6%的内存空间，但是匹配速率也只有 AC 算法的 45%左右，因而只适合于大规模模式串且模式串长度较短的应用场景。

b）基于哈希散列的匹配算法。这类算法是在 BM[9]单模式匹配领域的扩展，主要思路是使用散列来记录模式串的信息，并用字符块匹配来提高跳转幅度。比较有代表性的基于哈希散列的匹配算法包括 KR[10]算法和 WM 算法。这类算法的优点是空间占用小，在大字符集上匹配速度很快，但是当最短模式串长度较小或者模式串集合规模过大时匹配速度也会受到很大限制。针对上述问题，文献[11]引入了 LONG_SHIFT 和 SHORT_SHIFT 两张哈希表，用于在可能出现匹配时做进一步的校验，两张表分别进行长模式串检验和短模式串校验，目的是为了提高检验率，解决最短模式串长度较小而引起的跳转距离过小的问题。这种方法在模式串长度分布不均匀时能取得一定的效果，但对于模式串集合中短模式串较多的情况匹配的效率难以得到有效提升。文献[12]结合 QS 算法对 WM 算法进行了改进，提出了 QWM 算法，该算法在匹配阶段判断当前窗口匹配失败时检查当前窗口后面的 B 字符块，使得最大移动距离从 m-B+1 增大为 m+B。QWM 算法使用误匹配信息来增加匹配窗口的跳跃距离，在中文字符集上取得了 7%左右的提升效果，但是当模式串规模较大时匹配效率提高不大。

c）基于位并行的匹配算法。这类算法是用位图来模拟状态机的匹配过程，即用位向量运算替代状态机跳转，并用机器码来并行运算加快速度，有代表性的是 Shift-AND/Shift-OR[13]算法。这类算法空间占用小且匹配速率较高，但是受限于机器字节宽度的影响，通常只适用于小规模模式集的匹配。针对这个问题，文献[14]提出了结合 q-gram 技术来对 Shift-And 算法进行扩展，首先将所有模式串规约成一个通配模式串 p；然后用位并行算法 shift-And 对 p 进行搜索，对可能出现匹配的情况进行二级精确校验，从而极大地增加了校验率，但是这种方法在大于 10 万的超大规模模式集上仍然没有较好表现。

目前国内外学者针对多模式匹配算法进行大量研究，并提出了很多新的改进算法，尤其在大规模模式串的匹配方面有了很大的进展。文献[15]提出了 TFD 算法来处理大规模 URL 匹配的问题，结合散列技术和双数组 Trie 树加速精确校验的进程，同时优化了存储空间。但是 Trie 数据结构需要占用较大的存储空间，导致该算法的内存开销很大，限制了该算法的实际应用。文献[16]基于经典的 SOG 匹配算法，提出了一种过滤型的改进算法 SOGOPT。该算法对模式串集合进行分组并且为每个模式串选择了特定的窗口，大幅度地提高了匹配性能。但是由于机器字节宽度的影响，当最短模式串的长度较长时该算法的匹配性能较差。

尽管多模式匹配算法已经有很多研究成果，但是针对 PDF 文本内容的多模式匹配，尤其当模式集合较大时的研究成果还很少。本文在 Wu-Manber 算法的基础上，结合中文 PDF 文本内容的编码规则，从增大跳转幅度和减少无谓的精确校验次数的角度出发，采用双重哈希跳转和布隆过滤器等多种优化技术，提高了中文 PDF 文本的多模式匹配性能。

3　本文算法设计研究

本章提出一种适用于 PDF 文本内容审查的多模式匹配算法，该算法是在经典的 Wu-Manber 算法基础上，结合中文 PDF 文本编码规则的特点进行了改进，以适应模式串集合规模很大时的 PDF 文本匹配的需求。本文在 3.1 节中介绍了 Wu-Manber 算法，并分析了该算法在 PDF 文本匹配中的不足和缺陷。针对这些不足之处，在 3.2~3.4 节介绍了本文的三种改进方法。3.5 节是本文提出改进算法的具体描述，包括预处理和扫描匹配过程。

3.1　Wu-Manber 算法分析

Wu-Manber 算法是 Sun Wu 在 1994 年提出的经典多模式匹配算法，该算法利用了 BM 算法中提出的“坏字符”跳转的思想，是一种启发式的算法。为了方便描述，首先给出 Wu-Manber 算法的符号定义，如表 1 所示。

表 1　Wu-Manber 算法符号表

Table 1　Parameter description of Wu-Manber algorithm

符号	描述
Σ	字母集
m	模式集合最短长度
B	字符块大小
T	文本
table_size	SHIFT、PREFIX、HASH 表大小
k	模式串集合的数量
$X=\{x_1, x_2, \cdots, x_B\}$	长度为 B 的字符块
$P=\{p_1, p_2, \cdots, p_k\}$	模式集合，P_i 是第 i 个模式串

本节将对 Wu-Manber 算法的预处理和扫描阶段做具体阐述，并分析该算法在 PDF 文本匹配中的缺点。

3.1.1 预处理阶段

在预处理中，先根据模式串集合计算最短模式串长度 m，随后利用 m、k 和 Σ 计算字符块 B 和 table_size 的值，最后截取每个模式串的前 m 个字符作为匹配窗口来构造了三张哈希表：SHIFT、HASH 和 PREFIX 表。三张哈希表的构造方法如下：

a) SHIFT 表。SHIFT 用于存储任意长度为 B 的字符块 V 的哈希值，此处称为跳转值。如果 V 不存在于模式串集合的任何模式串中，则跳转值为 m-B+1；否则定义 V 出现在所有

模式串最右位置的索引值为 d，则跳转值为 m-d。式（1）为 SHIFT 表的计算方法。

$$SHIFT[V]=\begin{cases} m-B+1, V\notin P \\ \min\{m-d|\forall jPj[d-i]=V[B-i]\}, V\in P \end{cases} \quad (1)$$

b) HASH 表。HASH 表用于存储每个模式串前 m 个字符中最右端长度为 B 的字符块的哈希值，此处称为后缀哈希值。拥有相同后缀哈希值的字符块被放入同一个链表中存储。

c) PREFIX 表。PREFIX 表用于存储每个模式串前 m 个字符中最左端长度为 B 的字符块的哈希值，此处称为前缀哈希值。

3.1.2 扫描阶段

扫描阶段主要是利用之前构造好的三张哈希表，从前到后地扫描文本 T，并不断检测是否匹配到模式串，直到扫描到文本末尾。具体过程如下：

a) 在文本 T 中，计算当前匹配窗口中最右端长度为 B 的字符块的哈希值，记作 h。

b) 查看 SHIFT[h]的值，若大于 0 则匹配窗口按照 SHIFT[h]的值向右移动，执行步骤 a)；若等于 0 则执行步骤 c)。

c) 计算当前匹配窗口的最左端长度为 B 的字符块的哈希值，记做 pre_hash。

d) 将 HASH[h]所在链表的全部模式串，逐个计算 PREFIX[h]的值，判断是否等于 pre_hash。

e) 若不等于则跳过，回到步骤 d)。

f) 若等于，则对该模式串进行逐字符地精确匹配。

g) 让该模式串从当前匹配窗口的最左端开始从前到后地精确匹配。HASH[h]所在链表全部模式串都校验完成后，将匹配窗口向右移动 1 个距离，执行步骤 a)。

3.1.3 Wu-Manber 算法的缺点

当将 Wu-Manber 算法应用于模式串规模较大的 PDF 文本内容审查时匹配效率不佳，主要有两方面的原因，详细阐述如下：

a）跳转距离小，哈希冲突大。一方面，经典 Wu-Manber 算法中，字符块 B 的取值一般为 2 或 3，在 PDF 中文文本的内容检索中，一个汉字会对应 4 个字符，当最短模式串是 4 个汉字时，转换编码后 m 是 16，此时 B 的值若为 2 将会产生大量的哈希冲突而造成校验成功率下降，减缓了匹配进程；另一方面，Wu-Manber 算法中 SHIFT 表的最大跳转值为 m-B+1，且在精确校验后跳转值为 1，在 PDF 文本匹配中该跳转值明显可以进行增加且不会出现匹配遗漏。

b）精确检验耗时严重。Wu-Manber 算法在匹配过程中，当 SHIFT 表的跳转值为 0 时表明可能匹配到了模式串。此时就需要进行精确校验来判断是否匹配成功，而这个过程需要逐一地遍历对应 HASH 表映射的链表中的全部模式串。由于中文 PDF 文本采用 CID 编码格式，使得模式串的长度较长且相同前缀出现的比例较大，相同前缀的冲突链很长，导致在精确校验时遍历较多的模式串而造成 CPU 消耗很大，匹配效率低下。

3.2 调整 SHIFT 表跳转值

根据 PDF 中文文本编码具有四个字符组成一个汉字的特点，在预处理阶段，初始化 SHIFT 哈希表时键的取值可以相隔四个单位。此外，将 Wu-Manber 算法的 SHIFT 表默认跳转值（最大跳转值），即当文本串中的字符块没有出现在任何模式串中时的 SHIFT 跳转值，从 m-B+1 改为 m-B+4。在匹配阶段，精确校验结束后的跳转值，在 Wu-Manber 算法中固定为 1，然而该跳转值可以动态地得到，在本文 3.3 节中会具体描述该值的计算方法，并且由于 CID 编码的特殊性，最小跳转值为 4。

例如，对于模式串集合 P={不可思议，妙不可言，信息时代}，在某 PDF 中转换为 CID 编码后，P={4e0d53ef601d8bae,59994e0d53ef8a00,4fe1606f65f64ee3}，根据如下公式(2)计算字符块大小 B 的值：

$$B=\lceil 4*\log_{\Sigma}(2*m*r)\rceil \quad (2)$$

其中：Σ是字符集的大小，是模式串的个数。取字符块大小 B=8，则根据该模式串集合构造的 SHIFT 表，如表 2 所示。可以看出，每个字符块的跳转值都是 4 的倍数，因此加大了跳转的幅度。

表 2　SHIFT 表键值对映射关系

Table 2　Key-value pair mapping of SHIFT table

字符块	跳转值
4e0d53ef	4
53ef601d	4
601d8bae	0
59994e0d	8
53ef8a00	0
4a1606f6	8
0bf65f64	4
5f644ee3	0

3.3 双重哈希跳转

Wu-Manber 算法中，将匹配窗口的最右端大小为 B 的字符块命名为 H，则当 SHIFT[H]=0 时会检验 PREFIX 表之后进行精确校验，结束后匹配窗口仅向右滑动 1 个距离。当文本串规模较大时，精确校验次数很多而滑动距离只有 1 会导致算法的匹配效率不高。

为此，本文新增了一张 JUMP 哈希表，它的键与 HASH 表相同，而值表示精确匹配后的精确跳转值：当前验证的字符块 H 没有出现在其他模式串的匹配窗口时，JUMP[H]=m-B+4；当前验证的字符块 H 只在其他模式串的匹配窗口最左端出现时，JUMP[H]=m-B；其余情况下 JUMP[H]=4。因此，在匹配阶段结束精确校验后，跳转值根据 JUMP[H]得到，这样将匹配阶段的无谓时间消耗转移到预处理阶段，加快了匹配性能。

3.4 使用布隆过滤器减少无谓匹配

布隆过滤器(Bloom filter)是由 Burton Bloom [17] 在 1970 年提出的一种高效的随机数据存储结构，它用位图简洁地表示一个集合，并且快速地判断一个元素是否属于该集合。布隆过滤器是基于多哈希函数映射压缩参数空间的数据结构，将 k 个哈希函数运算后的值映射到位图中，空间和时间效率都很高 [18]。

由于 PDF 中文文本的编码格式，即一个汉字带有四个字符的信息，在匹配阶段进入精确匹配但是没有匹配到模式串的情况很多，这样造成精确匹配的成功率较低。为此，本文提出使用布隆过滤器来避免不需要进行精确匹配的情况，从而加速匹配。首先，定义当前匹配窗口的后缀字符块和下一个匹配窗口的前缀的字符块组成长度为 B 的新字符块为 HB。在预处理阶段，将所有非最短模式串的字符块 HB 加入到布隆过滤器中。在匹配阶段，当检测到可能出现匹配时，在查找 PREFIX 和 HASH 表之前计算字符块 HB 是否在布隆过滤器之中，如果存在则进行精确校验，否则直接根据 JUMP[H] 来跳转。

布隆过滤器可以快速地检索一个元素是否在集合中，它的优点是时间和空间效率都远超同类的算法，但是它也存在假阳性误判的缺陷。假阳性误判，即布隆过滤器有一定概率将不在集合中的元素误判为在集合中。假阳率的计算如下公式（3）所示。

$$f = (1 - e^{-\frac{nk}{s}})^{k} \tag{3}$$

其中：n 表示加入布隆过滤器的规则集数量；k 是布隆过滤器中哈希函数的数目；s 则表示布隆过滤器中位图的比特数量。由上式可以看出，适当地选择 s 和 k 的值可以降低假阳率。从式（3）可以推导出 n、s、k 三者的关系为

$$k = (\frac{s}{n}) In2 \tag{4}$$

其中：$\frac{s}{n}$ 表示布隆过滤器映射每个元素所需的比特位数。为了避免布隆过滤器的假阳性误判，本文选择两个明显独立的哈希函数[19]：SDBM 和 SAX。SDBM 在不同的数据集中有很好的差方分布，SAX 则具有高效的移位运算速度，两者的计算公式如下：

$$SDBM = c + (hash << 6) + (hash << 16) - hash \tag{5}$$

$$SAX = c + (hash << 5) + (hash >> 2) \tag{6}$$

3.5　改进算法描述

本文提出的改进算法包括预处理和扫描匹配阶段。预处理阶段主要包括根据模式集计算最短模式串长度 m 值，利用所有模式串匹配窗口的后缀字符块构造 HASH 和 JUMP 表，并根据匹配窗口的前缀字符块构造 PREFIX 表以及非后缀字符块构造 SHIFT 表，最后用 3.4 节提出的 HB 字符块构造布隆过滤器(Bloom Filter)。图 2 是预处理的示意图。其中灰色部分代表长度为 m 的匹配窗口，用于计算 SHIFT、HASH、PREFIX 和 JUMP 表。表中 index 表示由字符块计算出的哈希值，shift 和 skip 都表示跳转值，id 代表一个模式串的编号。深灰色部分代表 HB 字符块，用于初始化布隆过滤器。

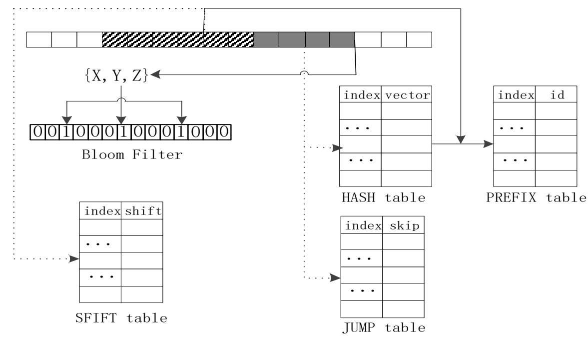


图 2　改进算法预处理示意图

Fig. 2　Schematic diagram of improved algorithm preprocessing

扫描匹配阶段会利用预处理生成的四张哈希表以及布隆过滤器，算法 1 的伪代码是对该阶段的描述。匹配过程中始终维持长度为 m 的匹配窗口向右移动，假设当前匹配窗口的后缀长度为 B 的字符块为 H，则判断 SHIFT[H]>0 时直接跳转 SHIFT[H]的值；否则对字符块 H 的后缀和下一匹配窗口的前缀组成的长度为 B 的字符块 HB 进行布隆过滤器校验，若验通过则进行 PREFIX 和 HASH 表的精确校验，否则直接跳过 JUMP[H]的值。

算法 1　扫描文本匹配算法

```
index←m-1, text_len←len(text)
while index < text_len do
block_hash←HashCode(text+index-B,B)
shift_value←SHIFT[block_hash]
if shift_value > 0 then
index←index + shift_value
else if not BloomFilter.contains(index,B) then
       index←index + JUMP[block_hash]
else
        prefix_hash←HashCode(text+index-m,B)
while HASH[prefix_hash]≠null do
             find match HASH[prefix_hash][id]
end while
index←index + JUMP[block_hash]
end if
end while
```

图 3 和 4 是改进算法和 Wu-Manber 算法处理相同实例的结果。其中模式集合是 P={新时代、曾几何时、算几何题目}，在某 PDF 中用 CID 编码转换后 P={079e85h679c1、3f89463d8a09237f、293b463d8a091c63 73a2}。由此可知，最短模式串长度 m=12，B=8，预处理阶段各表项的取值如图 3 所示。

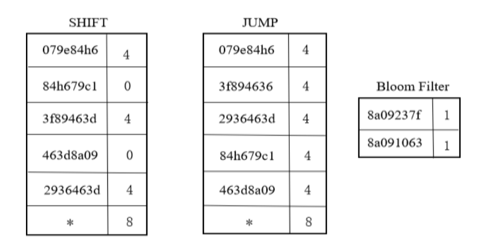


图 3　改进算法预处理阶段例子

Fig. 3　Example of improved algorithm preprocessing

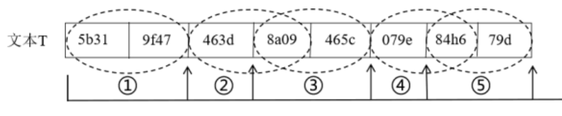


图 4　改进算法扫描匹配阶段例子

Fig. 4　Example of improved algorithm scan matching

图 4 显示了改进算法扫描匹配阶段的过程。当改进算法计算到匹配窗口字符 463d8a09 在 SHIFT 表中对应值为 0 时，不会立即进行精确校验，而是检测当前匹配窗口后缀字符块和下一个匹配窗口前缀字符块组成的长度为 B 的字符块 8a09465c 是否在布隆过滤器中，检测失败后文本向后移动 JUMP[463d8a09]=4 个距离，继续检查直到文本结束。

在这个例子中，相比于 Wu-Manber 算法，改进算法的平均跳跃距离（5.6 次）大于 WM 算法的平均跳跃距离（2.8 次），并且改进算法避免了进行一次精确校验，从而加速了匹配的进程。由此可知，改进算法在处理中文 PDF 文本时，当最短模式串长度较长且各个模式串中存在相同前后缀时，改进算法具有跳跃距离大和精确校验次数少的优势，从而使匹配的效率更高。

4　实验评估

本文的实验评估除了对算法整体的匹配速率进行对比之外，还考虑了 Wu-Manber 算法中精确匹配成功率（MSR）和哈希表跳转率（HSR）的比较。本文将匹配窗口右端字符块定义为 H，SHIFT[H]>0 以及被布隆过滤器拦截而直接跳转的次数记为 SkipCount，SHIFT[H]=0 时进入精确匹配的次数记为 MatchCount，成功匹配到模式串的次数记为 SuccessCount，

则精确匹配成功率（MSR）的和哈希表跳转率（HSR）的计算公式如式（7）（8）所示。

$$MSR = \frac{SuccessCount}{MatchCount} \quad (7)$$

$$HSR = \frac{SkipCount}{SkipCount + MatchCount} \quad (8)$$

由于 Wu-Manber 算法中在进入精确匹配后可能出现误匹配而减缓匹配性能，所以 MSR 的值越高说明算法效率越好。哈希表跳转率高则表明算法的哈希表访问率低，即哈希计算的次数少，所以 HSR 的值越高说明算法的时间效率越好。

4.1　实验数据和实验环境

本文使用的实验软硬环境是：Intel C612 E5-2600v4 CPU @ 3.20 GHz 16 核；内存 32 GB；操作系统为 Centos 6.6；内核版本是 3.10.0-123. e17.x86_64。程序代码使用 C++语言编写，g++ 5.4.0 编译，单线程运行。

本文实验中选取的 PDF 文档包括 100 个 100 KB~100 MB 的 PDF 文件。由于篇幅的限制，表 3 列出了部分实验样本。随后使用 PDFBinder 1.2 软件将 100 个 PDF 文档进行合并，构成大小约 500 MB 的 PDF 文件，最后提取其中的 stream 文本内容用于算法性能测试。为了充分评估算法性能，实验根据 PDF 文本内容随机生成了数目分别为 10 000、20 000、30 000、40 000、50 000、60 000、70 000、80 000 共 8 个模式集合，其中最短模式串为 12，最长模式串为 168。

表 3　实验的部分 PDF 文档

Table 3　PDF documents of experiment

文件名	文件大小
基于 PDF 文档的网络学习资源建设.pdf	112 KB
PDF1.7 文档规范.pdf	397 KB
一种改进的多模式匹配算法.pdf	530 KB
LWIP 协议栈源码详解.pdf	1481 KB
C 和指针.pdf	11328 KB
Hadoop 权威指南.pdf	51930 KB

4.2　实验结果及分析

首先对上文提到的精确匹配成功率（MSR）和哈希跳转率（HSR）进行了测试。实验选用的模式集规模为 10 000，在最短模式串递增的情况下对 WM 和改进算法分别进行了实验测试。图 5 和 6 是测试结果示意图。从图中可以看出，改进算法的 MSR 和 HSR 相比于传统的 WM 算法都有较大提升，这是因为随着最短模式串长度的增大，布隆过滤器利用模式串信息而过滤掉的无谓匹配更多，二重哈希的跳转次数和距离也随之增大，所以改进算法的 MSR 和 HSR 相比于 WM 算法提升明显。

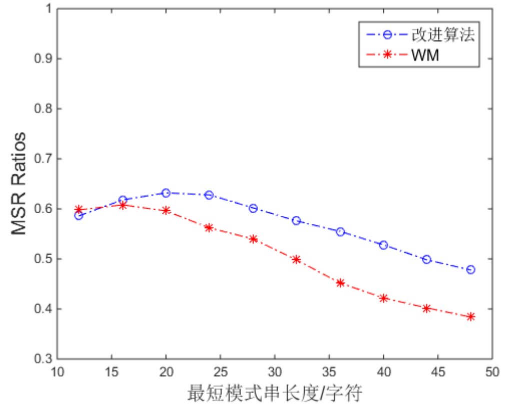


图 5　模式串规模为 10 000 时 MSR 对比

Fig. 5　MSR comparison when pattern string size of 10 000

图 7~9 是算法整体匹配速度的仿真。实验中模式串共有 10 000 条，模式串与文本串的占比为 17.62%。由图可见，使用本文提出的改进算法相比于其他算法在 PDF 文本的匹配上有更高的性能。当最短模式串长度为 12 时，本文提出的改进算法在模式串规模为万级时的匹配速度和 SOGOPT 算法相差不大，但相比于 WM 和 QWM 算法匹配速度可以提升大约为 20% ~40%。当最短模式串长度较长时，由于模式串所携带的信息更多，本文提出的改进算法中双重哈希跳转和布隆过滤器的作用得以体现，即匹配过程中跳转幅度更大，精确校验次数更少，从而使得匹配速度开始明显优于其他算法，尤其当模式串规模增大时匹配速度更是达到 QWM 和 WM 算法的一倍以上。而 SOGOPT 算法随着最短模式串长度和模式串规模的增大，它采用的分组规约受到限制而造成精确校验次数增多，导致其算法效率有所下降。

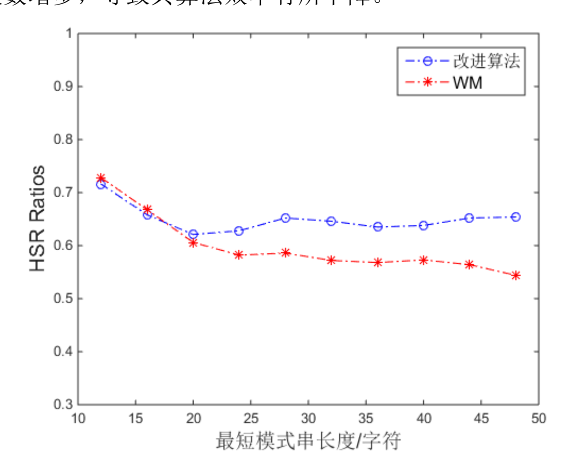


图 6　模式串规模为 10 000 时 HSR 对比

Fig. 6　HSR comparison when pattern string size of 10 000

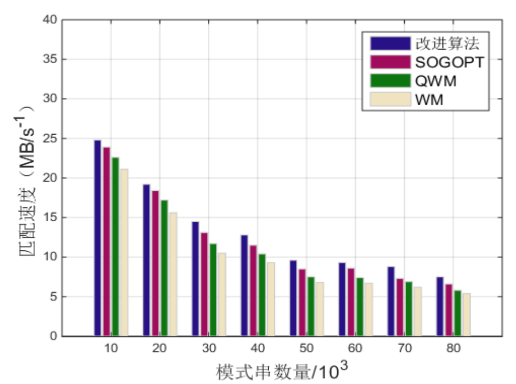


图 7　模式串最短长度为 12 时匹配速度对比

Fig. 7　Comparison of matching speed when shortest string length is 12

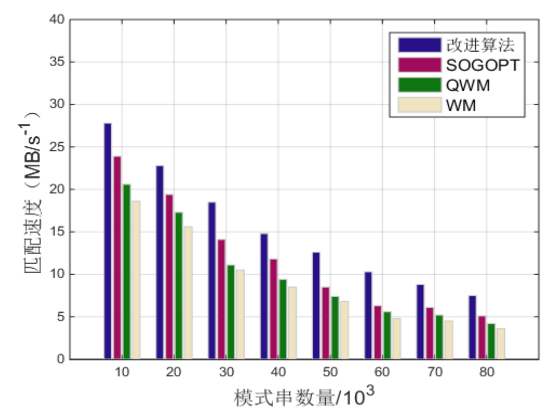


图 8　模式串最短长度为 16 时匹配速度对比

Fig. 8　Comparison of matching speed when shortest string length is 16

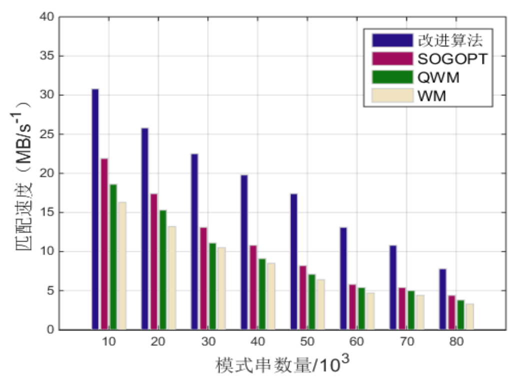


图 9　模式串最短长度为 20 时匹配速度对比

Fig. 9　Comparison of matching speed when shortest string length is 20

5　结束语

在互联网流量爆炸的今天，PDF 文件的内容监管和过滤在网络信息安全领域中愈发重要。本文分析了 Wu-Manber 算法在 PDF 文本匹配中的缺点和不足，并根据 PDF 内容流的编码规则，利用双重哈希跳转和布隆过滤器优化匹配性能，提出一种适用于 PDF 文本匹配的改进算法。在真实数据中测试表明，本文提出的改进算法和其他算法相比在中文 PDF 文本审查中匹配效率更高，尤其在大规模模式串的应用环境中性能更佳。

目前 PDF 文件的版本更新日益频繁，内容流的标签订位也在不同版本中有不同的方式。在今后的工作中，将研究不同版本 PDF 的中英文编码规范，结合本文及其他改进算法，希望研究出具有普遍适用能力的中英文 PDF 文本高效匹配算法。

参考文献：

[1] Stylianopoulos C, Almgren M, Landsiedel O. Multiple pattern matching for network security applications: acceleration through vectorization [C]// Proc of the 46th IEEE International Conference on Parallel Processing. Bristol: IEEE Press, 2017: 472-482.

[2] Langlois J M P, Savaria Y. Memory-efficient string matching for intrusion detection systems using a high-precision pattern grouping algorithm [C]// Proc of Symposium on Architectures for Networking and Communications Systems. California, USA: ACM Press, 2016: 37-42.

[3] 曹成宏，雷迎科，徐一鸣. 面向链路层比特流数据频繁统计的 AC-IM 算法 [J]. 小型微型计算机系统，2018, 39 (7): 62-66. (Cao Chenghong, Lei Yingke, Xu Yiming. AC-IM algorithm oriented frequent statistics of link-layer bit stream data [J]. Journal of Chinese Computer Systems, 2018, 39 (7): 62-66.)

[4] Tahir M, Sardaraz M, Ikram A A. EPMA: efficient pattern matching algorithm for DNA sequences [J]. Expert Systems with Applications, 2017, 80 (1): 162-170.

[5] 孙本阳，王轶骏，薛质. 一种改进的恶意 PDF 文档静态检测方案 [J]. 计算机应用与软件，2016, 33 (3): 308-313. (Sun Benyang, Wang Yijun, Xue Zhi. An improved static detection scheme for malicious pdf documents [J]. Computer Applications and Software, 2016, 33 (3): 308-313.)

[6] Wu S, Manber U. A fast algorithm for multi-pattern searching [J]. CiteSeer, 1994 (1): 1-20.

[7] Singh R, Mohaar G S. Fast document indexing using aho-corasick state machine [C]// Proc of the 17th IEEE International Conference on Information Reuse and Integration . New York: IEEE Press, 2016: 469-475.

[8] 张萍，刘燕兵，于静，等. HashTrie：一种空间高效的多模式串匹配算法 [J]. 通信学报，2015, 36 (10): 172-180. (Zhang Ping, Liu Yanbing, Yu Jing, *et al.* HashTrie: a space-efficient multiple string matching algorithm [J]. Journal on Communications, 2015, 36 (10): 172-180.)

[9] Rahim R, Ahmar A S, Ardyanti A P. Visual approach of searching process using Boyer-Moore algorithm [J]. Journal of Physics: Conference Series, 2017, 930 (1): 1-5.

[10] Karp R M, Rabin M O. Efficient randomized pattern-matching algorithms [J]. IBM J Res Dev, 1987, 31 (2): 249-260.

[11] 夏念，嵩天. 短规则有效的快速多模式匹配算法 [J]. 计算机工程与应用，2017, 53 (7): 1-8. (Xia Nian, Song Tian. Short rule efficient rapid multi-pattern matching algorithm [J]. Computer Engineering and Applications, 2017, 53 (7): 1-8.)

[12] Zhang W. An improved Wu-Manber multiple patterns matching algorithm [C]// Proc of IEEE Electronic Information and Communication Technology. New York: IEEE Press, 2016: 286-289.

[13] Saikrishna V, Rasool A, Khare N. Spam filtering through multiple pattern bit parallel string matching combining shift AND & OR [J]. International Journal of Computer Applications, 2013, 61 (5): 40-45.

[14] Elziky M A, Ibrahim D M, Sarhan A M. Improved duplicate record detection using ASCII code Q-gram indexing technique [J]. Arabian Journal for Science and Engineering, 2018, 43 (12): 7409-7420.

[15] Yuan Zhenlong, Yang Baohua, Ren Xiaoqi, *et al.* TFD: a multi pattern matching algorithm for large-scale URL filtering [C]// Proc of Computing, Networking and Communications. Beijing: IEEE Press, 2013: 359-363.

[16] 刘燕兵，邵妍，王勇，等. 一种面向大规模 URL 过滤的多模式串匹配算法 [J]. 计算机学报，2014, 37 (5): 1159-1169. (Liu Yanbing, Shao Yan, Wang Yong, *et al.* A multiple string matching algorithm for large-scale URL filtering [J]. Chinese Journal of Computers, 2014, 37 (5): 1159-1169.)

[17] Holley G, Wittler R, Stoye J. Bloom filter trie: an alignment-free and reference-free data structure for pan-genome storage [J]. Algorithms for Molecular Biology, 2016, 11 (1): 1-9.

[18] Aldwairi M, Al-Khamaiseh K. Exhaust: optimizing Wu-Manber pattern matching for intrusion detection using Bloom filters [C]// Proc of the 2nd World Symposium on Web Applications and Networking . Sousse, Tunisia: IEEE Press, 2015: 1-22.

[19] Chinaunix. 几种常用 hash 算法及原理 [EB/OL]. [2016-08-09]. http://blog. ChiNaunix. net/uid-30592332-id-5749402. html.